KSSOLV - A MATLAB Toolbox for Solving the Kohn-Sham Equations

Chao Yang and Juan C. Meza and Byounghak Lee and Lin-Wang Wang

We describe the design and implementation of KSSOLV, a MATLAB toolbox for solving a class of nonlinear eigenvalue problems known as the Kohn-Sham equations. These types of problems arise in electronic structure calculations, which are nowadays essential for studying the microscopic quantum mechanical properties of molecules, solids and other nanoscale materials. KSSOLV is well suited for developing new algorithms for solving the Kohn-Sham equations and is designed to enable researchers in computational and applied mathematics to investigate the convergence properties of the existing algorithms. The toolbox makes use of the object-oriented programming features available in MATLAB so that the process of setting up a physical system is straightforward and the amount of coding effort required to prototype, test and compare new algorithms is significantly reduced. All of these features should also make this package attractive to other computational scientists and students who wish to study small to medium size systems.

Categories and Subject Descriptors: G.1.10 [**Numerical Analysis**]: Applications – Electronic Structure Calculation; G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra; G.1.6 [**Numerical Analysis**]: Optimization; G. 4. [**Mathematics of Computing**]: Mathematical Software-Algorithm Design and Analysis

General Terms: nonlinear eigenvalue problem, density functional theory (DFT), Kohn-Sham equations, self-consistent field iteration (SCF), direct constrained minimization (DCM)

Additional Key Words and Phrases: planewave discretization, pseudopotential

1. INTRODUCTION

KSSOLV is a MATLAB toolbox for solving a class of nonlinear eigenvalue problems known as the Kohn-Sham equations. These types of problems arise in electronic structure calculations, which are nowadays essential for studying the microscopic quantum mechanical properties of molecules, solids and other nanoscale materials. Of the many approaches for studying the electronic structure of molecular systems, methods based on *Density Functional Theory* (DFT) [Hohenberg and Kohn 1964] have been shown to be among the most successful. Through the DFT formalism, one can reduce the many-body Schrödinger equation used to describe the electron-electron and electron-nucleus interactions to a set of single-electron equations that have far fewer degrees of freedom. These equations, which we will describe in more detail in the next section, were first developed by W. Kohn and L. J. Sham [Kohn and Sham 1965]. Discretizing the single-electron equations results in a set of nonlinear equations that resemble algebraic eigenvalue problems presented in standard linear algebra textbooks [Demmel 1997; Golub and Van Loan 1989; Trefethen and Bau III 1997]. The main feature distinguishing the Kohn-Sham equations from the standard linear eigenvalue problem is that the matrix operator in these equations is a function of the eigenvectors that must be computed. For this reason, the problem defined by the Kohn-Sham equations is more accurately described as a nonlinear eigenvalue problem.

Due to the nonlinear coupling between the matrix operator and its eigenvectors, the Kohn-Sham equations are more difficult to solve than standard linear eigenvalue problems. Currently, the most widely used numerical method for solving this type of problem is the Self Consistent Field (SCF) iteration, which we will examine in detail in Section 3. The SCF iteration has been implemented in almost all quantum chemistry and physics software packages. However, the mathematical convergence properties of SCF are not yet fully understood; for example, it is well known that the simplest form of SCF iteration often fails to converge to the correct solution [Le Bris 2005]. Although a number of techniques have been developed by chemists and physicists to improve the convergence of SCF, these methods are also not well understood, and they can fail in practice as well.

Clearly, more work is needed to investigate the mathematical properties of the Kohn-Sham equations, to rigorously analyze the convergence behavior of the SCF iteration, and to develop improved numerical methods that are both reliable and efficient. Some progress has recently been made in this direction [Le Bris 2005; Cancès and Le Bris 2000b; Cancès 2001]. However, many efforts have been hampered within the larger applied mathematics community by the lack of mathematical software tools that one can use to quickly grasp the numerical properties of the Kohn-Sham equations and to perform simple computational experiments on realistic systems.

The lack of such software tools also makes it difficult to introduce basic DFT concepts and algorithms into numerical analysis courses, even though these ideas are relatively well developed in computational chemistry and physics curricula. Although a number of well designed software packages are available for performing DFT calculations on large molecules and bulk systems [Gonze et al. 2002; Baroni et al. 2006; Kresse and Furthmüller 1996; Kronik et al. 2006; Andreoni and Curioni 2000; Wang 2008; Shao et al. 2006], it is often a daunting task for researchers and students with a minimal physics or chemistry background to delve into these codes to extract mathematical relations from various pieces of the software. Furthermore, because these codes are usually designed to handle large systems efficiently on parallel computers, the data structures employed to encode basic mathematical objects such as vectors and matrices are often sophisticated and difficult to understand. Consequently, standard numerical operations such as fast Fourier transforms, numerical quadrature calculations, and matrix vector multiplications become non-transparent, making it difficult for a computational mathematician to develop and test new ideas in such an environment.

The KSSOLV toolbox we developed provides a tool that will enable computational mathematicians and scientists to study properties of the Kohn-Sham equations by rapidly prototyping new algorithms and performing computational experiments more easily. It will also allow them to develop and compare numerical methods for solving these types of problems in a user friendly environment. One of the main features of KSSOLV is its objected-oriented design, which allows users with a minimal physics or chemistry background to assemble a realistic atomistic system quickly. The toolbox also allows developers to easily manipulate wavefunctions and Hamiltonians within a more familiar linear algebra framework.

We will present the main features and capabilities of KSSOLV in this paper. Since KSSOLV is targeted primarily towards users who are interested in the numerical analysis aspects of electronic structure calculations, our focus will be on numerical algorithms and how they can be easily prototyped within KSSOLV. We provide some background information on the Kohn-Sham equations and their properties in Section 2. Numerical methods for solving these types of problems are discussed in Section 3 along with some of the difficulties one may encounter. We then describe the design features and the implementation details of KSSOLV in Section 4. In Section 5, we illustrate how an algorithm for solving the Kohn-Sham equations can be easily implemented in KSSOLV. Several examples are provided in Section 6 to demonstrate how KSSOLV can be used to study the convergence behavior of different algorithms and visualize the computed results. Throughout this paper, we will use $\|\cdot\|$ to denote the 2-norm of a vector, and $\|\cdot\|_F$ to denote the Frobenius norm of a matrix.

2. KOHN-SHAM ENERGY MINIMIZATION

Properties of molecules, solids and other nanoscale materials are largely determined by the interactions among electrons in the outer shells of their atomic constituents. These interactions can be characterized quantitatively by the electron density, which can be viewed as a multi-dimensional probability distribution. The electron density of a many-atom system can be obtained by solving the well known many-body Schrödinger equation

$$H\Psi(r_1, r_2, ..., r_{n_e}) = \lambda\Psi(r_1, r_2, ..., r_{n_e}). \tag{1}$$

Here $\Psi(r_1, r_2, ..., r_{n_e})$ ($r_i \in \mathbb{R}^3$ and n_e is the number of electrons) is a many-body wavefunction whose magnitude squared characterizes an electronic configuration in a probabilistic sense, i.e., $|\Psi(r_1, r_2, ..., r_{n_e})|^2 dr_1 dr_2 \cdots dr_{n_e}$ represents the probability of finding electron 1 in a small volume around r_1, electron 2 in a small volume around r_2 etc., and

$$\int_\Omega \Psi^* \Psi d\Omega = 1, \tag{2}$$

where $\Omega = \Omega_1 \times \Omega_2 \cdots \Omega_{n_e}$, and $\Omega_i \subseteq \mathbb{R}^3$. Furthermore, the wavefunction must also obey the *antisymmetry principle*, defined by

$$\Psi(r_1, ..., r_i, ..., r_k, ..., r_{n_e}) = -\Psi(r_1, ..., r_k, ..., r_i, ..., r_{n_e}). \tag{3}$$

The differential operator H is a many-body Hamiltonian that relates the electronic configuration to the energy of the system. When appropriate boundary conditions are imposed, the energy must be quantized and is denoted here by $\lambda \in \mathbb{R}$.

Using the *Born-Oppenheimer* approximation, which is to say that we assume the positions of the nuclei $\hat{r}_j, j = 1, 2, ..., n_u$, are fixed, where n_u denotes the number of nuclei, the many-electron Hamiltonian H can be defined (in atomic units) by

$$H = -\frac{1}{2}\sum_{i=1}^{n_e}\Delta_{r_i} - \sum_{j=1}^{n_u}\sum_{i=1}^{n_e}\frac{z_j}{||r_i - \hat{r}_j||} + \sum_{1\leq i,j\leq n_e}\frac{1}{||r_i - r_j||}, \tag{4}$$

where Δ_{r_i} is the Laplacian operator associated with the ith electron, and z_j is the charge of the jth nucleus.

Equation (1) is clearly an eigenvalue problem. In many cases, we are interested in the eigenfunction Ψ associated with the smallest eigenvalue λ_1, which corresponds to the minimum (ground state) of the *total energy functional*

$$E_{total}(\Psi) = \int_{\Omega}\Psi^* H \Psi \quad d\Omega, \tag{5}$$

subject to the normalization and antisymmetry constraints (2) and (3). For atoms and small molecules that consist of a few electrons (less than three), we can discretize (1) and solve the eigenvalue problem directly. However, as n_e increases, the number of degrees of freedom in (1), after it is discretized, increases exponentially making the problem computationally intractable. For example, if r_i is discretized on an $m \times m \times m$ grid, the dimension of H is $n = m^{3n_e}$. For $m = 32$ and $n_e = 5$, n is greater than 3.5×10^{22}. Thus, it would be infeasible to solve the resulting eigenvalue problem on even the most powerful computers available today.

To address the dimensionality curse, several approximation techniques have been developed to decompose the many-body Schrödinger equation (1) into a set of single-electron equations that are coupled through the electron density (defined below). The most successful among these is based on *Density Functional Theory* [Hohenberg and Kohn 1964]. In their seminal work, Hohenberg and Kohn proved that at the ground-state, the total energy of an electronic system can be described completely by a function of the 3-D electron density

$$\rho(r) \equiv n_e \int_{\Omega\backslash\Omega_1} |\Psi(r, r_2, r_3, ..., r_{n_e})|^2 dr_2 dr_3 \cdots dr_{n_e}.$$

Assuming all electrons are indistinguishable, the quantity $\rho(r)dr/n_e$ gives the probability of finding an electron within a small volume around $r \in \mathbb{R}^3$.

Unfortunately the proof given in [Hohenberg and Kohn 1964] is not constructive and the analytical expression for this density-dependent total energy functional is unknown. Subsequently, Kohn and Sham [Kohn and Sham 1965] proposed a practical procedure to approximate the total energy by making use of single-electron wavefunctions associated with a non-interacting reference system. Using this Kohn-Sham model, the total energy (5) can be defined as

$$\begin{aligned} E_{total}^{KS} \;=\; & \frac{1}{2}\sum_{i=1}^{n_e}\int_{\Omega}||\nabla\psi_i(r)||^2 dr + \int_{\Omega}\rho(r)V_{ion}(r)dr + \\ & \frac{1}{2}\int_{\Omega}\int_{\Omega}\frac{\rho(r)\rho(r')}{||r-r'||}drdr' + E_{xc}(\rho), \end{aligned} \tag{6}$$

where ψ_i, $i = 1, 2, ..., n_e$ are the single-particle wavefunctions that satisfy the orthonormality constraint $\int \psi_i^* \psi_j = \delta_{i,j}$, and $\Omega \subset \mathbb{R}^3$. Here $\rho(r)$ is the *charge density* defined by

$$\rho(r) = \sum_{i=1}^{n_e} \psi_i^*(r)\psi_i(r). \tag{7}$$

The function $V_{ion}(r) = \sum_{j=1}^{n_u} z_j/||r - \hat{r}_j||$ represents the ionic potential induced by the nuclei, and $E_{xc}(\rho)$ is known as the exchange-correlation energy, which is a correction term used to account for energy that the non-interacting reference system fails to capture.

As the analytical form of $E_{xc}(\rho)$ is unknown, several approximations have been derived semi-empirically [Perdew and Zunger 1981; Perdew and Wang 1992]. In KSSOLV, we use the local density approximation (LDA) proposed in [Kohn and Sham 1965]. In particular, E_{xc} is expressed as

$$E_{xc}(\rho) = \int_\Omega \rho(r)\epsilon_{xc}[\rho(r)]dr, \tag{8}$$

where $\epsilon_{xc}(\rho)$ represents the exchange-correlation energy per particle in a uniform electron gas of density ρ. The analytical expression of ϵ_{xc} used in KSSOLV is the widely accepted formula developed in [Perdew and Zunger 1981]. To simplify the presentation, we have ignored the spin degree of freedom in $\psi_i(r)$, $\rho(r)$ and E_{xc}. For some applications, it is important to include this extra degree of freedom which gives the local spin-density approximation (LSDA) of E_{xc}.

It is not difficult to show that the first order necessary condition (Euler-Lagrange equation) for the constrained minimization problem

$$\begin{array}{ll} \min & E_{total}^{KS}(\{\psi_i\}) \\ \{\psi_i\} & \\ \text{s.t} & \psi_i^*\psi_j = \delta_{i,j} \end{array} \tag{9}$$

has the form

$$\begin{aligned} H(\rho)\psi_i &= \lambda_i\psi_i, \quad i = 1, 2, ..., n_e, && (10) \\ \psi_i^*\psi_j &= \delta_{i,j}. && (11) \end{aligned}$$

where the single-particle Hamiltonian $H(\rho)$ (also known as the Kohn-Sham Hamiltonian) is defined by

$$H(\rho) = -\frac{1}{2}\Delta + V_{ion}(r) + \rho(r) \star \frac{1}{||r||} + V_{xc}(\rho), \tag{12}$$

where $\star$ denotes the convolution operator. The function $V_{xc}(\rho)$ in (12) is the derivative of $E_{xc}(\rho)$ with respect to ρ. Because the Kohn-Sham Hamiltonian is a function of ρ, which is in turn a function of $\{\psi_i\}$, the set of equations defined by (10) results in a nonlinear eigenvalue problem. These equations are collectively referred to as the *Kohn-Sham equations*. Interested readers can learn more about these equations from several sources (e.g. [Nogueira et al. 2003]).

3. NUMERICAL METHODS

In this section, we will describe the numerical methods employed in KSSOLV to obtain an approximate solution to the Kohn-Sham equations (10)-(11). We begin by discussing the planewave discretization scheme that turns the continuous nonlinear problem into a finite-dimensional problem. The finite-dimensional problem is expressed as a matrix problem in Section 3.2. We present two different approaches to solving the matrix nonlinear eigenvalue problem in Sections 3.3 and 3.4. Both of these approaches have been implemented in KSSOLV.

3.1 Planewave Discretization

To solve the minimization problem (9) or the Kohn-Sham equations numerically, we must first discretize the continuous problem. Standard discretization schemes such as finite difference, finite elements and other basis expansion (Ritz-Galerkin) methods [Ritz 1908] all have been used in practice. The discretization scheme we have implemented in the current version of KSSOLV is a Ritz type of method that expresses a single electron wavefunction $\psi(r)$ as a linear combination of planewaves $\{e^{-ig_j^T r}\}$, where $g_j \in \mathbb{R}^3$ $(j = 1, 2, ..., n_g)$ are frequency vectors arranged in a lexicographical order. The planewave basis is a natural choice for studying periodic systems such as solids. It can also be applied to non-periodic structures (e.g., molecules) by embedding these structures in a fictitious supercell [Payne et al. 1992] that is periodically extended throughout an open domain. The use of the planewave basis has the additional advantage of making various energy calculations in density functional theory easy to implement. It is the most convenient choice for developing and testing numerical algorithms for solving the Kohn-Sham equations within the MATLAB environment, partly due to the availability of efficient fast Fourier transform (FFT) functions.

It is natural to assume that the potential for R-periodic atomistic systems is a periodic function with a period $R \equiv (R_1, R_2, R_3)$. Consequently, we can restrict ourselves to one canonical period often referred to as the primitive cell and impose periodic boundary conditions on the restricted problem. It follows from *Bloch*'s theorem [Ashcroft and Mermin 1976; Bloch 1928] that eigenfunctions of the restricted problem $\psi(r)$ can be periodically extended to the entire domain (to form the eigenfunction of the original Hamiltonian) by using the following formula:

$$\psi(r + R) = e^{ik^T R}\psi(r), \tag{13}$$

where $k = (k_1, k_2, k_3)$ is a frequency or wave vector that belongs to a primitive cell in the reciprocal space (e.g., the first *Brillouin* zone (BZ) [Ashcroft and Mermin 1976]). If the R-periodic system spans the entire infinite open domain, the set of k's allowed in (13) forms a continuum in the first Brillouin zone. That is, each $\psi(r)$ generates an infinite number of eigenfunctions for the periodic structure. It can be shown that the corresponding eigenvalues form a continuous cluster in the spectrum of the original Hamiltonian [Ashcroft and Mermin 1976]. Such a cluster is often referred to as an energy band in physics. Consequently, the complete set of eigenvectors of H can be indexed by the band number i and the Brillouin frequency vector k (often referred to as a k-point), i.e., $\psi_{i,k}$. In this case, the evaluation of the charge density must first be performed at each k-point by replacing $\psi_i(r)$ in (7) with $\psi_{i,k}$ to yield

$$\rho_k(r) = \sum_{i=1}^{n_e} |\psi_{i,k}(r)|^2.$$

The total charge density $\rho(r)$ can then be obtained by integrating over k, i.e.,

$$\rho(r) = \frac{|\Omega|}{(2\pi)^3} \int_{BZ} \rho_k(r) dk, \tag{14}$$

where $|\Omega|$ denotes the volume of the primitive cell in the first Brillouin zone. Furthermore, an integration with respect to k must also be performed for the kinetic energy term in (6).

When the primitive cell (or supercell) in real space is sufficiently large, the first Brillouin zone becomes so small that the integration with respect to k can be approximated by a single k-point calculation in (6) and (14).

To simplify our exposition, we will, from this point on, assume that a large primitive cell is chosen in the real space so that no integration with respect to k is necessary. Hence we will drop the index k in the following discussion and use $\psi(r)$ to represent an R-periodic single particle wavefunction. The periodic nature of $\psi(r)$ implies that it can be represented (under some mild assumptions) by a Fourier series, i.e.,

$$\psi(r) = \sum_{j=-\infty}^{\infty} c_j e^{i g_j^T r}, \tag{15}$$

where c_j is a Fourier coefficient that can be computed from

$$c_j = \int_{-R/2}^{R/2} \psi(r) e^{-i g_j^T r} dr.$$

To solve the Kohn-Sham equations numerically, the Fourier series expansion (15) must be truncated to allow a finite number of terms only. If all electrons are treated equally, the number of terms required in (15) will be extremely large. This is due to the observation that the strong interaction between a nucleus and the inner electrons of an atom, which can be attributed to the presence of singularity in $V_{ion}(r)$ at the the nuclei position $\hat{r}_j$, must be accounted for by high frequency planewaves. However, because the inner electrons are held tightly to the nuclei, they are not active in terms of chemical reactions, and they usually do not contribute to chemical bonding or other types of interaction among different atoms. On the other hand, the valence electrons (electrons in atomic orbits that are not completely filled) can be represented by a relatively small number of low frequency planewaves. These electrons are the most interesting ones to study because they are responsible for a majority of the physical properties of the atomistic system. Hence, it is natural to focus only on these valence electrons and treat the inner electrons as part of an ionic core. An approximation scheme that formalizes this approach is called the *pseudopotential* approximation [Phillips 1958; Phillips and Kleinman 1958; Yin and Cohen 1982]. The details of pseudopotential construction and their theoretical properties are beyond the scope of this paper. For the purpose of this paper, we shall just keep in mind that the use of pseudopotentials allows us to

(1) remove the singularity in V_{ion};

(2) reduce the number of electrons n_e in (6) and (7) to the number of valence electrons;

(3) represent the wavefunction associated with a valence electron by a small number of low frequency planewaves.

In practice, the exact number of terms used in (15) is determined by a kinetic energy cutoff E_{cut}. Such a cutoff yields an approximation

$$\psi(r) = \sum_{j=1}^{n_g} c_j e^{ig_j^T r}, \tag{16}$$

where n_g is chosen such that

$$||g_j||^2 < 2E_{cut}, \tag{17}$$

for all $j = 1, 2, \ldots, n_g$. Although, the value of n_g will depend on many parameters such as the size and type of the system being studied, it is typically an order of magnitude smaller than $n = n_1 \times n_2 \times n_3$.

Once E_{cut} is chosen, the minimal number of samples of r along each Cartesian coordinate direction (n_1, n_2, n_3) required to represent $\psi(r)$ (without the aliasing effect) can be determined from the sampling theorem [Nyquist 1928]. That is, we must choose n_k ($k = 1, 2, 3$) sufficiently large so that

$$\frac{1}{2}\left(\frac{2\pi n_k}{R_k}\right) > 2\sqrt{2E_{cut}}, \tag{18}$$

is satisfied, i.e., n_k must satisfy $n_k > 2R_k\sqrt{2E_{cut}}/\pi$.

We will denote the uniformly sampled $\psi(r)$ by a vector $x \in \mathbb{R}^n$, where $n = n_1n_2n_3$ and the Fourier coefficients c_j in (16) by a vector $c \in \mathbb{C}^n$ with zero paddings used to ensure the length of c matches that of x. If the elements of x and c are ordered properly, these two vectors satisfy

$$c = Fx. \tag{19}$$

where $F \in \mathbb{C}^{n\times n}$ is a discrete Fourier transform matrix [Van Loan 1987].

After a sampling grid has been properly defined, the approximation to the total energy can be evaluated by replacing the integrals in (6) and (8) with simple summations over the sampling grid.

The use of a planewave discretization makes it easy to evaluate the kinetic energy of the system. Since

$$\nabla_r e^{ig_j^T r} = ig_j e^{ig_j^T r},$$

the first term in (6) can be computed as

$$\frac{1}{2}\sum_{\ell=1}^{n_e}\sum_{j=1}^{n_g} ||g_j c_j^{(\ell)}||^2, \tag{20}$$

where $c_j^{(\ell)}$ is the jth Fourier coefficient of the wavefunction associated with the ℓth valence electron (denoted by x_ℓ). Here, one can take advantage of the orthogonality properties of the planewave basis, which allows one to remove the integral from the equation.

3.2 Finite-Dimensional Kohn-Sham Problem

If we let $X \equiv (x_1, x_2, ..., x_{n_e}) \in \mathbb{C}^{n \times n_e}$ be a matrix that contains n_e discretized wavefunctions, the approximation to the kinetic energy (6) can also be expressed by

$$\hat{E}_{kin} = \frac{1}{2}\text{trace}(X^* L X), \tag{21}$$

where L is a finite-dimensional representation of the Laplacian operator in the planewave basis. Due to the periodic boundary condition imposed in our problem, L is a block circulant matrix with circulant blocks that can be decomposed as

$$L = F^* D_g F, \tag{22}$$

where F is the discrete Fourier transform matrix used in (19), and D_g is a diagonal matrix with $||g_j||^2$ on the diagonal [Davis 1979]. If follows from (19) and (22) that (20) and (21) are equivalent.

In the planewave basis, the convolution that appears in the third term of (6) may be viewed as the $L^{-1}\rho(X)$, where $\rho(X) = \text{diag}(XX^*)$. (To simplify notation, we will drop X in $\rho(X)$ in the following.) However, since L is singular (due to the periodic boundary condition), its inverse does not exist. Similar singularity issues appear in the planewave representation of the pseudopotential and the calculation of the ion-ion interaction energy. However, it can be shown that the net effects of these singularities cancel out for a system that is electrically neutral [Ihm et al. 1979; Pickett 1989]. Thus, one can simply remove these singularities by replacing $L^{-1}\rho$ with $L^\dagger \rho$, where $L^\dagger$ is the pseudo-inverse of L defined as

$$L^\dagger = F^* D_g^\dagger F,$$

where $D_g^\dagger$ is a diagonal matrix whose diagonal entries (d_j) are

$$d_j = \begin{cases} ||g_j||^{-2} & \text{if } \; g_j \neq 0; \\ 0 & \text{otherwise.} \end{cases}$$

Consequently, the third term in (6), which corresponds to an approximation to the Coulomb potential, can be evaluated as

$$\hat{E}_{coul} = \rho^T L^\dagger \rho = [F\rho]^* D_g^\dagger [F\rho],$$

However, removing these singularities results in a constant shift of the total energy, for which a compensation must be made. It has been shown in [Ihm et al. 1979] that this compensation can be calculated by adding a term E_{rep} that measures the degree of repulsiveness of the local pseudopotential with a term that corresponds to the non-singular part of ion-ion potential energy. Because the second term can be evaluated efficiently by using a technique originally developed by Ewald [Ewald 1921], it is denoted by E_{Ewald}. Both E_{rep} and E_{Ewald} can be computed once and for all in a DFT calculation. We will not go into further details of how they are computed since they do not play any role in the algorithms we will examine in this paper.

To summarize, the use of a planewave basis allows us to define a finite-dimensional approximation to the total energy functional (6) as

$$\hat{E}_{total}(X) = \mathrm{trace}[X^*(\frac{1}{2}L + \hat{V}_{ion})X] + \frac{1}{2}\rho^T L^\dagger \rho + \rho^T \epsilon_{xc}(\rho) + E_{\mathrm{Ewald}} + E_{\mathrm{rep}}, \quad (23)$$

where $\hat{V}_{ion}$ denotes the ionic pseudopotentials sampled on the suitably chosen Cartesian grid of size $n_1 \times n_2 \times n_3$.

It is easy to verify that the KKT condition associated with the constrained minimization problem

$$\min_{X^*X=I} \hat{E}_{total}(X) \quad (24)$$

is

$$\begin{aligned} H(X)X - X\Lambda_{n_e} &= 0, \\ X^*X &= I, \end{aligned} \quad (25)$$

where

$$H(X) = \frac{1}{2}L + \hat{V}_{ion} + \mathrm{Diag}(L^\dagger \rho) + \mathrm{Diag}(\mu_{xc}(\rho)), \quad (26)$$

$\mu_{xc}(\rho) = d\epsilon_{xc}(\rho)/d\rho$, and Λ_{n_e} is a $n_e \times n_e$ symmetric matrix of Lagrangian multipliers. For simplicity, we will frequently denote the last three terms in (26) by

$$V_{tot} = \hat{V}_{ion} + \mathrm{Diag}(L^\dagger \rho) + \mathrm{Diag}(\mu_{xc}(\rho)), \quad (27)$$

Because $\hat{E}_{total}(X) = \hat{E}_{total}(XQ)$ for any orthogonal matrix $Q \in \mathbb{C}^{n_e \times n_e}$, we can always choose a particular Q such that Λ_{n_e} is diagonal. In this case, Λ_{n_e} contains n_e eigenvalues of $H(X)$. We are interested in the n_e smallest eigenvalues and the invariant subspace X associated with these eigenvalues.

3.3 The SCF Iteration

Currently, the most widely used algorithm for solving (25) is the self-consistent field (SCF) iteration which we outline in Figure 1 for completeness.

```
SCF Iteration

Input:    An initial guess of the wavefunction $X^{(0)} \in \mathbb{C}^{n \times n_e}$, pseudopotential;
Output:   $X \in \mathbb{C}^{n \times n_e}$ such that $X^*X = I_{n_e}$ and columns of $X$ span the invariant
          subspace associated with the smallest $n_e$ eigenvalues of $H(X)$ defined in (26).

1.        for $k = 1, 2, \ldots$ until convergence
2.           Form $H^{(k)} = H(X^{(k-1)})$;
3.           Compute $X^{(k)}$ such that $H^{(k)}X^{(k)} = X^{(k)}\Lambda^{(k)}$, and $\Lambda^{(k)}$
             contains the $n_e$ smallest eigenvalues of $H^{(k)}$;
4.        end for
```

Fig. 1. The SCF iteration

In [Yang et al. 2007], we viewed the SCF iteration as an indirect way to minimize $\hat{E}_{total}$ through the minimization of a sequence of quadratic surrogate functions of the form

$$q(X) = \frac{1}{2}\text{trace}(X^* H^{(k)} X), \tag{28}$$

on the manifold $X^*X = I_{n_e}$. This constrained minimization problem is solved in KSSOLV by running a small number of locally optimal preconditioned conjugate gradient (LOBPCG) iterations [Knyazev 2001].

Since the surrogate functions share the same gradient with $\hat{E}_{total}$ at $X^{(k)}$, i.e.,

$$\nabla \hat{E}_{total}(X)_{|X=X^{(k)}} = H^{(k)} X^{(k)} = \nabla q(X)_{|X=X^{(k)}},$$

moving along a descent direction associated with $q(X)$ is likely to produce a reduction in $\hat{E}_{total}$. However, because gradient information is local, there is no guarantee that the minimizer of $q(X)$, which may be far from $X^{(k)}$, will yield a lower $\hat{E}_{total}$ value. This observation partially explains why SCF often fails to converge. It also suggests at least two ways to improve the convergence of SCF.

One possible improvement is to replace the simple gradient-matching surrogate $q(X)$ with another quadratic function whose minimizer is more likely to yield a reduction in $\hat{E}_{total}$. In practice, this alternative quadratic function is often constructed by replacing the charge density $\rho^{(k)}$ in (26) with a linear combination of m previously computed charge densities, i.e.,

$$\rho_{mix} = \sum_{j=0}^{m-1} \alpha_j \rho^{(k-j)},$$

where $a = (\alpha_0, \alpha_2, ..., \alpha_{k-m+1})$ is chosen as the solution to the following minimization problem:

$$\min_{a^T e=1} \|Ra\|^2 \tag{29}$$

where $R = (\Delta\rho^{(k)} \;\; \Delta\rho^{(k-1)} \;\; ... \;\; \Delta\rho^{(m-1)}), \Delta\rho^{(k)} = \rho^{(k)} - \rho^{(k-1)}$ and $e = (1, 1, \ldots, 1)^T$. This technique is often called *charge mixing*. The particular mixing scheme defined by the solution to (29) is called *Pulay mixing* because it was first proposed by Pulay for Hartree-Fock calculations [Pulay 1980; 1982]. (In computational chemistry, Pulay mixing is referred to as the method of *direct inversion of iterative subspace* or simply DIIS). Other mixing schemes include *Kerker mixing* [Kerker 1981], *Thomas-Fermi mixing* [Raczkowski et al. 2001] and *Broyden mixing* [Kresse and Furthmüller 1996]. Charge mixing is often quite effective in practice for improving the convergence SCF even though its convergence properties are still not well understood. In some cases, charge mixing may fail also [Cancès and Le Bris 2000a; Yang et al. 2005].

Another way to improve the convergence of the SCF iteration is to impose an additional constraint to the surrogate minimization problem (28) so that the wavefunction update can be restricted within a small neighborhood of the gradient matching point $X^{(k)}$, thereby ensuring a reduction of the total energy function as we minimize the surrogate function. In [Yang et al. 2007], we showed that the following type of constraint

$$\|XX^* - X^{(k)} X^{(k)^*}\|_F^2 \leq \Delta,$$

where $\Delta > 0$ is a suitably chosen parameter, is preferred because it is rotationally invariant (i.e., post-multiplying X by an unitary matrix does not change the constraint) and because adding such a constraint does not increase the complexity of solving the surrogate minimization problem. It is not difficult to show [Yang et al. 2007] that solving the following constrained minimization problem,

$$\begin{array}{c} \min q(X) \\ XX^* = I \\ \|XX^* - X^{(k)}X^{(k)^*}\|_F^2 \le \Delta \end{array} \tag{30}$$

is equivalent to solving a low rank perturbed linear eigenvalue problem

$$\left[H^{(k)} - \sigma X^{(k)}X^{(k)^*}\right]X = X\Lambda, \tag{31}$$

where σ can be viewed as the Lagrange multiplier for the inequality constraint in (30), and Λ is a diagonal matrix that contains the n_e smallest eigenvalues of the low-rank perturbed matrix $H^{(k)}$. When σ is sufficiently large (which corresponds to a trust region radius Δ that is sufficiently small), the solution to (31) is guaranteed to produce a reduction in $\hat{E}_{total}(X)$.

When n_e is relatively small compared to n, the computational complexity of the SCF iteration is dominated by the floating point operations carried out in the multiplications of $H^{(k)}$ with discretized wavefunctions in X. These multiplications are performed repeatedly in an iterative method (e.g., the LOBPCG method or the Lanczos method) used at Step 3 in Figure 1 to obtain an approximate minimizer of (28). When a planewave expansion is used to represent X, each multiplication requires the use of a 3-D FFT operation to convert the Fourier space representation of each column of X into the real space representation before multiplications involving local potential terms in (27) can be carried out. An inverse 3-D FFT is required to convert the product back to the Fourier space. The complexity of each conversion is $\mathcal{O}(n \log n)$. If m LOBPCG iterations are used on average to obtain an approximate minimizer of (28), the total number 3-D FFTs required per SCF iteration is $2mn_e$. In addition, each SCF iteration also performs $\mathcal{O}(n \cdot n_e^2)$ basic linear algebra (BLAS) operations. When n_e becomes larger, these operations can become a significant part of the computational cost.

The amount of memory required by the SCF iteration consists of $3n_g n_e$ double precision and complex arithmetic words that must be allocated to store the current approximation to the desired wavefunctions, the gradient of the total energy, and additional workspace required in the LOBPCG or Lanczos algorithm for eigenvector calculations. An additional γn words are needed to store the various potential components in the Hamiltonian, the charge density approximation ρ as well as vectors that must be saved to perform charge mixing in (29), where the value of γ is typically less than 20.

3.4 Direct Constrained Minimization

Instead of focusing on Kohn-Sham equations (25) and minimizing the total energy indirectly in the SCF iteration, we can minimize the total energy directly in an iterative procedure that involves finding a sequence of search directions along which $\hat{E}_{total}(X)$ decreases and computing an appropriate step length. In most of the earlier direct minimization methods developed in [Arias et al. 1992; Gillan 1989; Kresse and Furthmüller 1996; Payne et al. 1992; Teter et al. 1989; VandeVondele and Hutter 2003; Voorhis and Head-Gordon 2002], the search direction and step length computations are carried out separately. This separation sometimes results in slow convergence. We recently developed a new direct constrained minimization (DCM) algorithm [Yang et al. 2005; 2007] in which the search direction and step length are obtained simultaneously in each iteration by minimizing the total energy within a subspace spanned by columns of

$$Y = \left(X^{(k)}, M^{-1}R^{(k)}, P^{(k-1)} \right),$$

where $X^{(k)}$ is the approximation to X obtained at the kth iteration, $R^{(k)} = H^{(k)}X^{(k)} - X^{(k)}\Lambda^{(k)}$, M is a Hermitian positive definite preconditioner, and $P^{(k-1)}$ is the search direction obtained in the previous iteration. It was shown in [Yang et al. 2005] that solving the subspace minimization problem is equivalent to computing the eigenvectors G associated with the n_e smallest eigenvalues of the following nonlinear eigenvalue problem

$$\hat{H}(G)G = BG\Omega, \quad G^*BG = I, \tag{32}$$

where

$$\hat{H}(G) = Y^* \left[\frac{1}{2}L + V_{ion} + \mathrm{Diag}\Big(L^{\dagger}\rho(YG) \Big) + \mathrm{Diag}\Big(\mu_{xc}(\rho(YG)) \Big) \right] Y, \tag{33}$$

and $B = Y^*Y$.

Because the dimension of $\hat{H}(G)$ is $3n_e \times 3n_e$, which is normally much smaller than that of $H(X)$, it is relatively easy to solve (32) by, for example, a trust region enabled SCF (TRSCF) iteration. We should note that it is not necessary to solve (32) to full accuracy in the early stage of the DCM algorithm because all we need is a G that yields sufficient reduction in the objective function.

Once G is obtained, we can update the wave function by

$$X^{(k+1)} \leftarrow YG.$$

The search direction associated with this update is defined, using the MATLAB submatrix notation, to be

$$P^{(k)} \equiv Y(:, n_e + 1 : 3n_e)G(n_e + 1 : 3n_e, :).$$

A complete description of the constrained minimization algorithm is shown in Figure 2. We should point out that solving the projected optimization problem in Step 7 of the algorithm requires us to evaluate the projected Hamiltonian $\hat{H}(G)$ repeatedly as we search for the best G. However, since the first two terms of $\hat{H}$ do not depend on G. They can be computed and stored in advance. Only the last two terms of (33) need to be updated. These updates require the charge density, the Coulomb and the exchange-correlation potentials to be recomputed.

In each DCM iteration, n_e Hamiltonian-wavefunction multiplications are performed to obtain the gradient. When an iterative method is used to solved the projected nonlinear eigenvalue problem (32), the charge density $\rho(YG)$ and the projected Hamiltonian must be updated repeatedly. The update of the projected

```
Algorithm: A Constrained Minimization Algorithm for Total Energy Minimization

Input:  initial set of wave functions $X^{(0)} \in \mathbb{C}^{n \times n_e}$; ionic pseudopotential; a preconditioner $M$;
Output: $X \in \mathbb{C}^{n \times k}$ such that the Kohn-Sham total energy functional $E_{total}(X)$ is minimized and $X^*X = I_k$.
1.   Orthonormalize $X^{(0)}$ such that $X^{(0)*}X^{(0)} = I_k$;
2.   for $k = 0, 1, 2, \ldots$ until convergence
3.       Compute $\Theta = X^{(k)*}H^{(k)}X^{(k)}$;
4.       Compute $R = H^{(k)}X^{(k)} - X^{(k)}\Theta$,
5.       if $(i > 1)$ then
             $Y \leftarrow (X^{(k)}, M^{-1}R, P^{(k-1)})$
         else
             $Y \leftarrow (X^{(k)}, M^{-1}R)$;
         endif
6.       $B \leftarrow Y^*Y$;
7.       Find $G \in \mathbb{C}^{2n_e \times n_e}$ or $\mathbb{C}^{3n_e \times n_e}$ that minimizes $E_{total}(YG)$
         subject to the constraint $G^*BG = I_{n_e}$;
8.       Set $X^{(k+1)} = YG$;
9.       if $(i > 1)$ then
             $P^{(k)} \leftarrow Y(:, n_e + 1 : 3n_e)G(n_e + 1 : 3n_e, :)$;
         else
             $P^{(k)} \leftarrow Y(:, n_e + 1 : 2n_e)G(n_e + 1 : 2n_e, :)$;
         endif
10.  end for
```

Fig. 2. A Direct Constrained Minimization Algorithm for Total Energy Minimization

Hartree potential requires us to compute $L^{\dagger}\rho(YG)$. This calculation makes use of two 3-D FFTs, hence has a complexity of $\mathcal{O}(n \log n)$. If m inner iterations are taken in the DCM algorithm to solve the projected problem, the total number of 3-D FFTs used per DCM iteration is $2(n_e + m)$. The memory requirement of the DCM algorithm is similar to that of an SCF iteration.

4. THE OBJECT-ORIENTED DESIGN OF KSSOLV

Both the SCF iteration and the DCM algorithm have been implemented in the KSSOLV toolbox, which is written entirely in MATLAB. It is designed to be modular, hierarchical, and extensible so that other algorithms can be easily developed under the same framework. In addition to taking advantage of efficient linear algebra operations and the 3-D fast Fourier transform (FFT) function available in MATLAB, the toolbox also makes use of MATLAB's object-oriented programming (OOP) features. KSSOLV contains several predefined classes that can be easily used to build a physical atomistic model in MATLAB and to construct numerical objects associated with planewave discretized Kohn-Sham equations. These classes are listed in Table I. The class names that appear in the first column of this table are treated as keywords in KSSOLV. We will demonstrate how specific instances of these classes (called objects) are created and used in KSSOLV. The internal structure of these classes are explained in detail in [Yang 2007].

The use of the object-oriented design allows us to achieve two main objectives:

(1) Simplify the process of setting up a molecular or bulk system and converting physical attributes of the system to numerical objects that users can work with easily.
(2) Enable numerical analysts and computational scientists to easily develop, test and compare different algorithms for solving the Kohn-Sham equation.

Class name	Purpose
`Atom`	Defines attributes of an atom
`Molecule`	Defines attributes of a molecule or a basic cell of a periodic system
`Hamilt`	Defines attributes of a Kohn-Sham Hamiltonian, e.g, potential
`Wavefun`	Defines one or a set of wavefunctions
`FreqMask`	Defines a mask used to filter high frequency components of a wavefunction

Table I. Classes defined in KSSOLV

In the following, we will illustrate how to define a molecular or bulk system in KSSOLV by creating `Atom` and `Molecule` objects. We will then show how to set up a Kohn-Sham Hamiltonian, which is represented as a `Hamilt` object, associated with a `Molecule` object. In KSSOLV, 3-D wavefunctions are represented as `Wavefun` objects. Although each `Wavefun` object stores the Fourier coefficients of a truncated planewave expansion of one or a few wavefunctions in a compact way, it can be manipulated as either a vector or a matrix. Both the `Hamilt` and the `Wavefun` objects are used extensively in the KSSOLV implementation of the SCF and DCM algorithms. As we will see in the following, using these objects significantly reduces the coding effort required to implement or prototype numerical algorithms for solving the Kohn-Sham equations.

4.1 From Atoms to Molecules and Crystals

To solve the Kohn-Sham equations associated with a particular molecular or bulk system in KSSOLV, we must first construct a `Molecule` object. Even though a bulk system (such as a crystal) is physically different from a molecule, we currently do not make such a distinction in KSSOLV. Both systems are considered periodic. In the case of a molecule, the periodicity is introduced by placing the molecule in a fictitious supercell that is periodically extended.

To construct a `Molecule` object, we use

```
mol = Molecule();
```

to first create an empty object called `mol` (a user-defined variable name). This call simply sets up the required data structure that is used to describe attributes of `mol`.

Before `mol` can be used in subsequent calculations, we must initialize all of its essential attributes, which include the number and type of atoms in this molecule, the size and shape of the supercell that contains the molecule, etc. All these attributes can be defined by using the `set` method associated with the `Molecule` class. The syntax of the `set` function is

```
mol = set(mol,attrname,attrvalue);
```

where the input argument `attrname` is a predefined string associated with the `Molecule` class that gives the name of a particular attribute, and `attrvalue` is a user supplied quantity that will be stored in `mol`. Table II lists the essential attributes that must be defined before the `mol` object can be used in subsequent calculations.

attribute name	purpose	value type
supercell	the primitive or super cell that contains the basic atomic constituents	a 3×3 matrix
atomlist	list of atoms	an array of `Atom` objects
xyzlist	list of atomic coordinates	an $n_a \times 3$ matrix, where n_a denotes the number of atoms
ecut	kinetic energy cutoff used for planewave discretization	a scalar

Table II. Attributes to be set in a `Molecule` object

Each molecule consists of a number of atoms. An atom is defined as an `Atom` object using the `Atom` constructor and a parameter that denotes either its chemical symbol or its atomic number. For example

```
a = Atom('Si')
```

or

```
a = Atom(14)
```

defines a silicon atom object named `a`. The `Atom` constructor internally calculates the number of valence electrons (n_e) and retrieves the shell configuration through a table lookup that finds the electron orbitals associated with each atom.

Atoms within a molecule can be placed in an array to form a list that is used to specify the `atomlist` attribute of a `Molecule` object. For example, a silane molecule contains one silicon (Si) atom and four hydrogen (H) atoms. After declaring both the Si and H atoms through the commands

```
a1 = Atom('Si')
a2 = Atom('H')
```

we can form the atom list associated with this molecule by

```
alist = [a1; a2; a2; a2; a2].
```

The list can then be used to define the `atomlist` attribute of the `mol` object by

```
mol = set(mol,'atomlist',alist).
```

To complete the description of the atomic configuration of a molecule, we must also specify the spatial location of each atom. In KSSOLV, the 3-D coordinates of the atoms can be placed in an $n_a \times 3$ matrix and passed to a `Molecule` object by setting the `xyzlist` attribute. For example, the atomic coordinates associated with the atoms in SiH_4 shown in Figure 3 is set by using

```
xyzmat = [  0      0       0
           1.61  1.61   1.61
          -1.61 -1.61   1.61
           1.61 -1.61  -1.61
          -1.61  1.61  -1.61 ];
mol = set(mol,'xyzlist',xyzmat).
```

Each row of the `xyzmat` array gives the (x, y, z) coordinates (in atomic/Bohr units) of the corresponding atom in the `alist` array defined above. Care must be taken in the input of these coordinates as KSSOLV does only a minimal checking for physically reasonable coordinates.

In addition to physical properties of a molecule or bulk system, the definition of a `Molecule` object in KSSOLV must also contain discretization information. The two main attributes that affect the discretization of the molecular system are the size and orientation of the supercell and the kinetic energy cutoff. Although these attributes are not properties of the physical system, including them in the `Molecule` class simplifies the construction of the Hamiltonian object and subsequent calculations.

The `supercell` attribute defines the shape and size of the primitive or supercell that contains the basic atomic constituents of a crystal or molecule. In KSSOLV, the supercell is described by a 3×3 matrix. Each column of this matrix defines the direction and length of one particular edge of the cell (or a translation vector) emanating from the origin. For example,

```
mol = set(mol,'supercell',10*eye(3));
```

sets the supercell of `mol` to a $10 \times 10 \times 10$ cube (in atomic units) whose three edges are parallel to the x, y and z axes respectively.

The attribute `ecut` is used to specify the kinetic energy cutoff that determines the number of effective planewave basis functions (n_g) and spatial sampling points (n_1, n_2 and n_3) used in the discretization. A higher cutoff energy leads to the use of a larger number of planewave basis functions and more spatial sampling grid points. This often yields a more accurate finite-dimensional approximation at the expense of higher computational cost. The optimal energy cutoff will depend on the molecular system being studied and the choice of pseudopotentials used in the Hamiltonian.

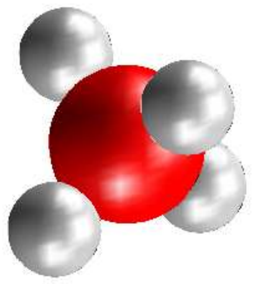

Fig. 3. The relative positions of all atoms in the SiH_4 molecule.

4.2 The Hamiltonian Class

A properly defined `Molecule` object `mol` can be used to initialize the Kohn-Sham Hamiltonian associated with this object. The initialization can be done by calling the constructor

```
H = Hamilt(mol).
```

Although the Kohn-Sham Hamiltonian $H(X)$ is treated as a matrix in equation (25), it is not stored as a matrix in KSSOLV. Instead, the `Hamilt` class keeps L, $\hat{V}_{ion}$ and the total potential $V_{tot} = \hat{V}_{ion} + \text{Diag}(L^{\dagger}\rho) + \text{Diag}(\mu_{xc}(\rho))$ as separate attributes. This separation makes it easier to update the Hamiltonian in both the SCF and the DCM calculations.

The kinetic component of a `Hamilt` object contains a compact representation of the frequency vectors g_j $(j = 1, 2, ..., n_g)$ that satisfy (17). These frequency vectors correspond to the nonzero diagonal elements of D_g in (22). The compact representation stores both the numerical values of $||g_j||^2$ and their (x, y, z) locations in a full 3-D representation. High frequency vectors that do not meet the criterion (17) are treated as zeros and never used when the kinetic component of the Kohn-Sham Hamiltonian is applied to a wavefunction.

The ionic potential $\hat{V}_{ion}$ contains a local term that is constructed and stored as a 3-D array at initialization. The nonlocal portion of $\hat{V}_{ion}$ is a low rank linear operator. In the frequency space, it can be represented as WW^* for some $n_g \times \ell$ $(\ell \ll n_g)$ matrix W, where ℓ is total number of relevant atomic orbitals (i.e., s, p, d orbitals) associated with all atoms in the molecule object. The construction of both the local and nonlocal portions of the ionic potentials makes use of the numerical procedure developed by Kleinman and Bylander [Kleinman and Bylander 1982] to sum up the atomic pseudopotentials stored in the KSSOLV subdirectory `Pseudopot`. KSSOLV provides Troullier-Martins [Troullier and Martins 1991] atomic pseudopotentials associated with most elements in the first four rows of the periodic table as well as a few commonly used elements in higher rows.

The determination of both the Coulomb and exchange correlation potential requires the availability of the charge density ρ which is in turn a function of the wavefunctions to be computed. Since a good approximation to the desired wavefunctions is not available at initialization, the initial ρ is computed in KSSOLV by combining atomic charge densities associated with each atom in the `mol` object.

In addition to standard potentials that appear within the Kohn-Sham density functional formalism, KSSOLV allows a user to specify other external potentials that electrons may experience through the '`vext`' attribute of a `Hamilt` object. All of these are stored internally as 3-D arrays.

Once a `Hamilt` object has been defined, one can retrieve various attributes of the object through the `get` function, e.g.,

```
vt = get(H,'vtot');
```

returns the total potential from a `Hamilt` object named `H` and assigns it to a user defined variable `vt`. This potential, which is stored as a MATLAB 3-D array, can be used and updated in a subsequent SCF or DCM calculation. An updated `vt` can be passed back into `H` by using the `set` function

```
H = set(H,'vtot',vt);
```

4.3 The Wavefunction Class

We created a special class `Wavefun` in KSSOLV to represent one or a set of wavefunctions. The creation of this class enables users to manipulate wavefunctions as if they are vectors or matrices. Additionally, since by construction the wavefunctions are represented by planewaves whose frequencies satisfy (17), we can use a compact data structure to reduce the cost of performing linear algebra operations.

In KSSOLV, a `Wavefun` object can be constructed using either a noncompact scheme or a compact scheme. In a noncompact representation, a `Wavefun` object `X` can be constructed through the command

```
X = Wavefun(psi),
```

where `psi` is a MATLAB 3-D array if `X` represents a single wavefunction, or a cell array that contains a list of 3-D arrays if `X` represents a set of wavefunctions.

Under the compact scheme, a `Wavefun` object stores only the nonzero Fourier expansion coefficients of a single wavefunction $\psi(r)$ or a set of wavefunctions $\{\psi_i(r)\}_{i=1}^{n_e}$. These coefficients are stored contiguously as a MATLAB cell array of size n_g by n_e. Such a storage scheme makes linear algebra operations on `X` more efficient. However, to perform 3-D FFT operations on `X`, we must place these coefficients in a 3-D array. Hence, in the compact scheme, we must also record the locations of these nonzero Fourier coefficients. Because the magnitude of each nonzero Fourier coefficient is smaller than a cutoff frequency as determined from (17), these coefficients lie within a sphere of a given radius in the 3-D Fourier space. In signal processing, functions whose Fourier coefficients satisfy a constraint of the type (17) are called *bandlimited* functions. In KSSOLV, the locations of the non-zero Fourier coefficients is stored in a separate array which is labeled as the '`idxnz`' attribute of the object. The '`n1`', '`n2`' and '`n3`' attributes, which gives the dimension of the wavefunction in real space, must be properly set in this case before the object can be used in subsequent calculation.

In the following section, we will see that all major matrix operations have been overloaded for `Wavefun` objects for both the compact and noncompact schemes. KSSOLV also provides a utility function `genX0` that allows one to easily construct initial `Wavefun` objects for the SCF or DCM calculations. To generate a set of random bandlimited wavefunctions using the kinetic energy cutoff specified in a `Molecule` object `mol`, one can simply use the command

```
X = genX0(mol).
```

Converting a `Wavefun` object `X` to a 3-D array (or a list of 3-D arrays) is straightforward when `X` is constructed using a noncompact scheme. The following command

```
X3D = get(X,'psi')
```

returns the wavefunctions as a cell array `X3D` of 3-D arrays. Although rarely needed when using KSSOLV, the following lines of codes show how the same conversion can be accomplished for an `X` constructed using a compact representation scheme

```
n1  = get(X,'n1');
```

```
n2  = get(X,'n2');
n3  = get(X,'n3');
psi = get(X,'psi');
idx = get(X,'idxnz');
X3D = zeros(n1,n2,n3);
X3D(idx) = psi{1}.
```

4.4 Operator Overloading

Because the Kohn-Sham Hamiltonian $H(X)$ and wavefunction X are viewed as matrices in (25), it is desirable to allow `Hamilt` and `Wavefun` objects to be manipulated in KSSOLV as if they are matrices. This feature is made possible in KSSOLV by overloading some basic algebraic operations for a `Wavefun` object. These overloaded operations are listed in Table III. One should be careful about the use of some of these operators. For example, since the wavefunctions used in the SCF and DCM calculation all have the same dimension, the multiplication operator `*` is never used between two `Wavefun` objects except when the first `Wavefun` object is transposed or conjugate transposed, i.e., it is valid to perform `x'*y` or `x.'*y`, and the multiplication returns a standard MATLAB matrix object. The overloaded multiplication operator `*` for `Wavefun` objects allows the second operand to be a standard matrix object with proper dimension. The result of the multiplication is a `Wavefun` object.

Operations	Description
`x + y`	Add two wavefunctions
`x - y`	Subtract one wavefunction from another
`x * y`	Multiply two wavefunctions and return a matrix
`x * a`	Multiply several wavefunctions with a matrix
`x .* y`	Element-wise multiplication of two wavefunctions
`x . y`	Element-wise division of two wavefunctions
`x'`	Complex conjugate transpose of a wavefunction
`x.'`	Transpose of a wavefunction
`[x y]`	Horizontal concatenation of several wavefunctions
`x(:,i:j)`	Subscripted reference of wavefunctions

Table III. Overload operations for `Wavefun` objects in KSSOLV

The multiplication operator is also overloaded for the `Hamilt` class so that the multiplication of a `Hamilt` object `H` and a `Wavefun` object `X` can be accomplished in KSSOLV by a simple expression

```
Y = H*X,
```

which hides all the complexity of the operation from the user.

4.5 Solvers

KSSOLV provides implementations of both the SCF and DCM algorithms for solving the Kohn-Sham equations. It also contains an implementation of the LOBPCG [Knyazev 2001] algorithm that can be used to compute a few of the smallest eigenvalues and the corresponding eigenvectors associated with a fixed Hamiltonian. The names of these solvers and their functionality are briefly described in Table IV.

Solver	Description
`scf.m`	An implementation of the SCF iteration with charge mixing.
`dcm.m`	An implementation of the DCM algorithm without trust region.
`trdcm1.m`	An implementation of a trust region enabled DCM algorithm with a fixed trust region radius.
`trdcm.m`	An implementation of a trust region enabled DCM algorithm with an adaptive trust region radius.
`lobpcg.m`	An implementation of the LOBPCG algorithm for computing approximations to the smallest eigenvalues and the corresponding eigenvectors of a fixed Hamiltonian. (It is used by `scf.m`)
`lanczos.m`	A simple implementation of the Lanczos algorithm with full orthogonalization.
`chebyscf.m`	A simple implementation of a polynomial filtered SCF iteration proposed in [Bekas et al. 2005; Zhou et al. 2006].

Table IV. Solvers provided in KSSOLV

These solvers serve two purposes. First, they allow users to solve the Kohn-Sham equations associated with different atomistic systems and observe how existing methods perform. In addition, they also provide templates for users to base new algorithms on.

The simplest use of the `scf` and `dcm` functions are

```
[Etotvec, X, vtot, rho] = scf(mol)
[Etotvec, X, vtot, rho] = dcm(mol),
```

where `mol` is a `Molecule` object. Both of these functions return a vector of total energy (`Etotvec`) values computed at each iteration, the final approximation to the desired wavefunctions `X`, and the total potential `vtot` and charge density `rho` associated with `X`. A number of optional parameters can be passed into these functions to improve the efficiency of the computation or the quality of the solution. The parameters used in `scf` are listed in Table V along with their default values. A similar set of parameters for the `dcm` function can be found in the user's guide [Yang 2007]. These parameters can be reset by passing a string-value pair as arguments to the `scf` or `dcm` function. For example,

```
[Etotvec, X, vtot, rho] = scf(mol,'pulaymix','off');
```

turns off the Pulay charge mixing scheme in the SCF iteration.

parameter name	purpose	default
`maxscfiter`	the maximum of SCF iterations allowed	10
`scftol`	the convergence tolerance for SCF	10^{-6}
`cgtol`	the convergence tolerance for the LOBPCG algorithm used to solve the linear eigenvalue problem in SCF	10^{-6}
`maxcgiter`	the maximum number of LOBPCG iterations allowed	10
`X0`	the starting guess of the wavefunction	random
`pulaymix`	activation of the Pulay charge mixing	'on'
`kerkmix`	activation of the Kerker charge mixing	'on'
`verbose`	detailed diagonostic printout	'on'

Table V. Parameters for SCF

By default, both the `scf` and `dcm` functions print out a list of diagnostic information on the screen. For example, Figure 4 shows the standard output from the `scf` function, which include eigenvalue approximations and residuals of the linear eigenvalue problem computed at each LOBPCG iteration as well as the approximate total energy, the 2-norm of each column of (25), and the difference between the input and output potentials computed at the end of each SCF iteration. These intermediate output can be used to monitor the convergence of the algorithm.

```
[Etotvec, X, vtot, rho] = scf(mol);
initialization...
Beging SCF calculation...
LOBPCG iter =   1
eigval( 1) =   7.195e+00, resnrm =   3.277e+00
eigval( 2) =   7.368e+00, resnrm =   3.347e+00
eigval( 3) =   7.551e+00, resnrm =   3.233e+00
eigval( 4) =   7.703e+00, resnrm =   3.239e+00

LOBPCG iter =   2
eigval( 1) =   5.097e-01, resnrm =   8.297e-01
eigval( 2) =   6.200e-01, resnrm =   8.719e-01
eigval( 3) =   6.912e-01, resnrm =   9.045e-01
eigval( 4) =   9.038e-01, resnrm =   9.130e-01
...
SCF iter  1:
norm(vout-vin) =  5.807e+00
Total energy   = -5.8719608972592e+00
 resnrm =   1.857e-02
 resnrm =   2.161e-02
 resnrm =   2.161e-02
 resnrm =   2.162e-02
...
```

Fig. 4. SCF output

5. ALGORITHM DEVELOPMENT UNDER KSSOLV

The use of an object-oriented design in KSSOLV simplifies the process of algorithm prototyping so that new algorithms can be implemented, tested and compared quickly. This is possible because many basic linear algebra operations can be applied directly to `Hamilt` and `Wavefun` objects. To give an example, we will show how the DCM algorithm can be easily translated into the MATLAB code shown in Figure 5. We should point out that the code segment shown in Figure 5 is a simplified version of the `dcm.m` file included in KSSOLV. The simplification is made to emphasize the main features of algorithm and its implementation.

In this example, a `Hamilt` object `H` has been constructed during an initialization step which is not shown here. A set of wavefunctions contained in a `Wavefun` object `X` has been created also. The code segment contained in the `while` loop constitutes a single DCM iteration. In this version of the DCM implementation, a simple, iteration count based termination criterion is used, i.e., the DCM iteration is terminated when the total number of DCM iterations reaches a user specified parameter `maxdcmiter`.

The first few lines of the codes within the `while` loop compute the preconditioned and gradient of the Kohn-Sham total energy (projected onto the tangent of the orthonormality constraint) with respect to the wave function `X`. They correspond to Steps 3 and 4 in Figure 2. The preconditioner `prec` used here is constructed using the techniques developed in [Teter et al. 1989]. In matrix notation, the preconditioner defined in [Teter et al. 1989] is diagonal in the frequency space. Thus it can be constructed as a `Wavefun` object, and the application of `prec` to wavefunctions stored in `R` can be carried out using an overloaded element-wise multiplication operation. The product is another `Wavefun` object.

We use the overloaded horizontal concatenation operator

```
Y = [X R];
if (iterdcm > 1) Y = [Y P]; end;
```

to construct the space spanned by wavefunctions contained in `X`, `R` and `P`. This matches exactly with Step 5 in Figure 2.

The MATLAB code in Figure 5 illustrates how the Kohn-Sham Hamiltonian is projected into the subspace spanned by wavefunctions contained in `Y` and how the projected problem is solved in DCM. The kinetic and ionic potential component of the Kohn-Sham Hamiltonian are projected outside of the inner SCF `for` loop used to solve the projected problem defined in Step 7 of the DCM algorithm. The function `applyKIEP`, which we do not show here, simply performs the operation $KY = (L + V_{ion})Y$, where KY is represented as a `Wavefun` object. The overloaded `Wavefun` multiplication operator makes the calculation $T = Y^*KY$ and $B = Y^*Y$ extremely easy.

The projection of the Coulomb and exchange-correlation potential must be done inside the inner SCF for loop because these nonlinear potentials change as eigenvectors of the projected Hamiltonian $\hat{H}(G)$ defined in (32) are updated. The projection is done by first computing $VY = [\mathrm{Diag}(L^{\dagger}\rho)) + \mathrm{Diag}(\mu_{xc}(\rho))]Y$ using the function `applyNP` (which we do not show here) and then performing a `Wavefunc` multiplication to obtain Y^*VY. The projected nonlinear potential is combined with the `T` matrix computed outside of the `for` loop to form the projected Kohn-Sham Hamiltonian `A`.

Because the dimensions of `A` and `B` are relatively small, we can compute all eigenvalues and the corresponding eigenvectors of the matrix pencil (`A`,`B`) in each inner SCF iteration using MATLAB's `eig` function. The returned eigenvalues and eigenvectors are sorted so that the leading `nocc` columns of `G` contain the eigenvectors associated with the `nocc` smallest eigenvalues of (`A`,`B`). These eigenvectors are used to update the wavefunctions `X` by multiplying `Y` with `G(:,1:nocc)` using the overload wavefunction multiplication operator. These calculations are followed by the update of the charge density `rho` as well as the recalculation of the Coulomb and exchange-correlation potentials and energies. The new nonlinear potential are then used to update the Hamiltonian by calling the `set` function.

6. EXAMPLES

The KSSOLV toolbox includes a number of examples that users can experiment with. Each example represents a particular molecule or bulk system. The system is created in a setup file. Table VI shows the names of all setup files and a brief

```
while ( iterdcm <= maxdcmiter )
   HX = H*X;
   T = X'*HX;
   R = HX - X*T; % the gradient of the total energy
   for j = 1:nocc
      R(:,j)=prec.*R(:,j); % apply the preconditioner prec
   end;
   % construct the projection subspace
   Y = [X R];
   if (iterdcm > 1) Y = [Y P]; end;
   ny = get(Y,'ncols');
   % project the kinetic and ionic potential part of the Hamiltonian into Y
   KY = applyKIEP(H,Y);  % KY = (L + Vion) Y
   T = Y'*KY;
   B = Y'*Y;
   % solve the projected problem
   G = eye(ny);
   for iterscf = 1:maxscfiter
      VY = applyNP(H,Y); %  project the nonlinear potential part of the Hamiltonian
      A = T + Y'*VY;
      [G,D]=eig(A,B,'chol');
      X = Y*G(:,1:nocc);
      rho = getcharge(mol,X,spintype);
      %  update the the Coulomb and exchange correlation potential
      [vcoul,vxc,uxc2,rho]=getvhxc(mol,rho);
      % recalculate kinetic, Coulomb and exchange-correlation and total energy
      Ekin  = (2/spintype)*trace( G(:,1:nocc)'*T*G(:,1:nocc) );
      Ecoul = getEcoul(mol,rho,vcoul);
      Exc   = getExc(mol,rho,uxc2);
      Etot  = Ewald + Ealphat + Ekin + Ecoul + Exc;
      % Update the nonlinear potential only
      H = set(H,'vnp',vcoul+vxc);
   end;
   % update the total potential
   vout = getvtot(mol, vion, vext, vcoul, vxc);
   H = set(H,'vtot',vout);
   % save the current "search direction"
   P = Y(:,nocc+1:ny)*G(nocc+1:ny,1:nocc);
   iterdcm = iterdcm + 1;
end;
```

Fig. 5. DCM in KSSOLV

description for each of them. It also shows the number of occupied states (`nocc`) which is simply the number of electron pairs for most systems (with the exception of the quantum dot example in which electrons are not paired by their spin orientations). To create a new system, a user can simply take one of the existing setup files and modify the construction of the `Molecule` object. KSSOLV provides atomic information (e.g., the atomic shell configuration in terms of s, p, d orbitals and the number of valence electrons etc.) and pseudopotentials associated with most elements in the first four rows of the periodic table as well as a few commonly used elements in higher rows. It can be used to study electronic properties of most small molecules (with up to a few hundred of valence electrons), insulators and semiconductor materials. For metals, a finite temperature formulation of the Kohn-Sham DFT model [Mermin 1965; Weinert and Davenport 1992; Wntzcovitch et al. 1992], which has not been implemented in current version of KSSOLV, must be used to generate physically meaningful results. We will add such a feature in the near future. In general, to produce physically meaningful results, appropriate supercell size and kinetic energy cutoff should be used. However, a user can experiment with different choices of these parameters to examine changes in the convergence properties of the numerical method or the quality of the computed solution in KSSOLV.

setup file name	nocc	Description
`c2h6_setup.m`	7	an ethane molecule
`co2_setup.m`	8	a carbon dioxide molecule
`h2o_setup.m`	4	a water molecule
`hnco_setup.m`	8	an isocyanic acid molecule
`qdot_setup.m`	8	an 8-electron quantum dot confined by external potential
`si2h4_setup.m`	6	a planar singlet silylene molecule
`sibulk_setup.m`	6	a silicon bulk system
`sih4_setup.m`	4	a silane molecule
`ptnio_setup.m`	43	a Pt_2Ni_6O molecule
`pentacene_setup.m`	102	a pentacene ($C_{22}H_{16}$) molecule

Table VI. Setup files for examples included in KSSOLV

Table VII shows that running an example shown in Table VI typically takes less than a minute on a Linux workstation, with the exception of the Pt_2Ni_6O and pentacene examples which took more than a few minutes to complete 10 SCF iterations. The timing results reported in the table were obtained on a single 2.2 GHz AMD Opteron processor. We used MATLAB Version 7.6.0.324 (R2008a) for all experiments reported in this paper. The total amount of memory available on the machine is 4 gigabytes (GB). A kinetic energy cutoff of 25 Ryd was used for most systems. For a $10\times10\times10$ (atomic units) cubic supercell, such a cutoff results in a $32 \times 32 \times 32$ sampling grid for the wavefunctions. For the Pt_2Ni_6O and the pentacene systems, the use of supercells requires the grid sizes to be increased to $63\times34\times30$ and $64\times32\times48$ respectively. Moreover, because the number of electrons in these systems are relatively large ($nocc = 43$ for Pt_2Ni_6O and $nocc = 102$ for pentacene), these problems took more time to solve.

The same initial guesses to the wavefunctions were used for both the SCF and DCM runs. All runs reported in the table used the default parameters in SCF and DCM. For example, in the case of SCF, the maximum number of LOBPCG iterations for solving each linear eigenvalue was set to 10. In the case of DCM, three inner SCF iterations were performed to obtain an approximate solution to (32).

The SCF and DCM errors reported in the last two columns of the table are the residual norms defined as

$$error = \|H(X)X - X\Lambda\|_F,$$

where X contains the wavefunctions returned from SCF or DCM and $\Lambda = X^*H(X)X$.

Table VII shows that DCM is faster than SCF for almost all systems. With the exception of the quantum dot example, it also produces more accurate results.

system	SCF time	DCM time	SCF error	DCM error
C_2H_6	26	25	9.4e-6	3.5e-6
CO_2	26	23	3.1e-3	1.1e-4
H_2O	16	16	5.7e-5	2.2e-5
$HNCO$	34	32	7.4e-3	6.8e-5
Quantum dot	18	16	5.0e-3	3.7e-1
Si_2H_4	25	23	1.8e-3	2.7e-4
silicon bulk	15	15	3.0e-4	9.6e-6
SiH_4	20	19	9.7e-6	4.9e-7
Pt_2Ni_6O	415	281	3.7	4.9e-2
pentacene	887	493	5.2e-1	2.5e-2

Table VII. Comparing the timing and accuracy of running SCF and DCM in KSSOLV

We will now take a closer look at two specific examples that demonstrate how KSSOLV can be used to solve the Kohn-Sham equations associated with different types of systems and how the computed results can be examined, compared and visualized in the MATLAB envrionment.

6.1 The Silane Molecule

The simplest example included in KSSOLV is perhaps the SiH_4 (silane) example described in the `sih4_setup.m`. The setup file contains the code snippets shown in Figure 6. These codes are used to construct a `Molecule` object for a molecule that consists of a silicon atom and four hydrogen atoms. The geometry configuration of these atoms is shown in Figure 3.

The overloaded `display` function for the `Molecule` class in KSSOLV allows users to see various attributes of the `mol` object by simply typing `mol` on the command line (without a semicolon at the end). Figure 7 shows the typical information one would see after typing `mol` on the command line.

The convergence behavior associated with different algorithms can be easily visualized by plotting the history of total energy reduction `Etotvec-Emin` where `Emin` is the minimum total energy computed by all methods. For example, Figure 8 shows how the total energy changes at each SCF and DCM iteration. We can clearly see from this figure that the reduction in total energy is more rapid in DCM than that in SCF for the silane system. Under the MATLAB environment, a user can easily modify the `scf` or `dcm` function to record and plot the change in total energy or Kohn-Sham residual norm with respect to either CPU time or the number of matrix vector multiplications performed. Similarly, we can compare the performance of the same algorithm with different parameter settings quite easily also. Figure 9 shows the use of charge mixing clearly accelerates the convergence of the SCF iteration.

For computational scientists, it is important to be able to examine the computed solution visually so that they may gain new insights into physical properties of the atomistic system under study. The MATLAB visualization capabilities make this task extremely easy. In Figure 10, we show the isosurface rendering of the charge density `rho` returned from the `scf` function. This figure is generated in MATLAB by using the command

```
% 1. construct atoms
a1 = Atom('Si');
a2 = Atom('H');
alist = [a1; a2; a2; a2; a2];
% 2. set up supercell
C = 10*eye(3);
% 3. define the coordinates the atoms
coefs = [
 0.0     0.0      0.0
 0.161   0.161    0.161
-0.161  -0.161    0.161
 0.161  -0.161   -0.161
-0.161   0.161   -0.161
];
xyzmat = coefs*C';
% 4. Configure the molecule
mol = Molecule();
mol = set(mol,'supercell',C);
mol = set(mol,'atomlist',alist);
mol = set(mol,'xyzlist' ,xyzmat);
mol = set(mol,'ecut', 25);  % kinetic energy cut off
mol = set(mol,'name','SiH4');
```

Fig. 6. Setting up the *Silane* molecule

```
>> mol
Molecule: SiH4
   supercell:
      1.000e+01     0.000e+00     0.000e+00
      0.000e+00     1.000e+01     0.000e+00
      0.000e+00     0.000e+00     1.000e+01
   sampling size: n1 = 32, n2 = 32, n3 = 32
   atoms and coordinates:
       1   Si      0.000e+00      0.000e+00      0.000e+00
       2    H      1.610e+00      1.610e+00      1.610e+00
       3    H     -1.610e+00     -1.610e+00      1.610e+00
       4    H      1.610e+00     -1.610e+00     -1.610e+00
       5    H     -1.610e+00      1.610e+00     -1.610e+00
   number of electrons  : 8
   spin type            : 1
   kinetic energy cutoff: 2.500e+01
```

Fig. 7. Displaying the attributes of the Silane molecule

```
isosurface(rho);
```

6.2 Electron quantum dot confined by an external potential

In addition to molecules and bulk systems, KSSOLV can be used to study the properties of quantum dots that consist of only electrons confined by an external potential field. The setup file `qdot_setup.m` which we list in Figure 11 shows how such a system can be created. Notice that no atomic information is needed in the setup file. Instead, we specify the number of electrons and set the spin

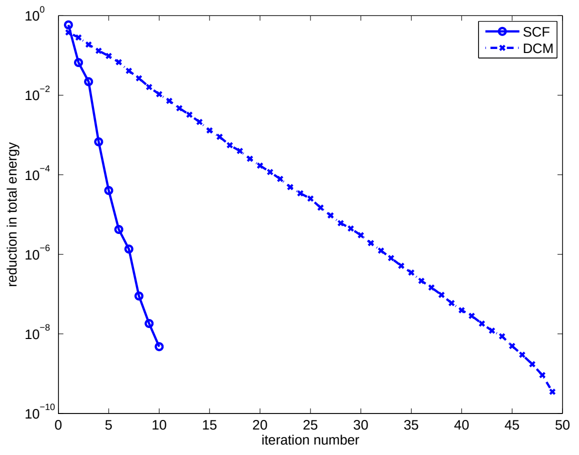

Fig. 8. Comparing the reduction of total energy in SCF and DCM for SiH_4.

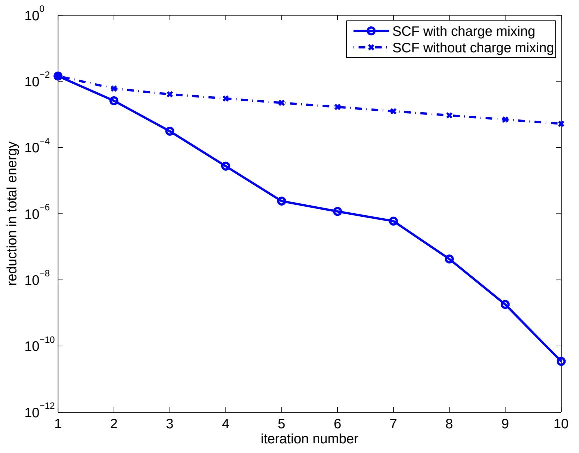

Fig. 9. The effect of charge mixing in SCF.

type (`spintype`) to 2, which indicates that the wavefunction associated with each electron is treated differently. The `getVharmonic` function used in the setup file (which we do not show here) defines an external potential parameterized by a parameter which is set to 1 in the setup file.

Figure 12 shows the attributes of the quantum dot when we type `mol` at the MATLAB prompt. Notice that the `atoms and coordinates` attribute is set to `none`. Figure 13 provides a partial listing of the output produced from running

```
[Etotvec, X, vtot, rho] = dcm(mol,'maxdcmiter',50);
```

The output shows that the convergence of DCM algorithm appears to be slow for this problem. In particular, the total energy changes very little from one DCM iteration to another. In the last few DCM iterations, there is no discernable change in total energy within the inner SCF iteration used to solve the projected problem.

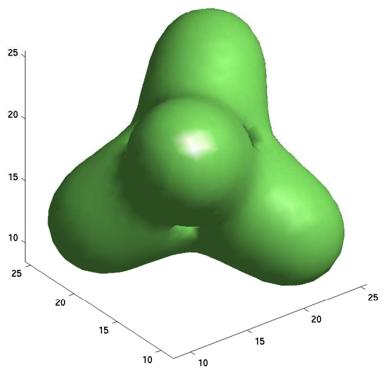

Fig. 10. The computed charge density of the SiH_4 molecule.

```
% 1. set up supercell
C = 10*eye(3);
% 2. Configure the molecule (crystal)
mol = Molecule();
mol = set(mol,'supercell',C);
mol = set(mol,'ecut', 25);  % kinetic energy cut off
mol = set(mol,'name','Quantum dot');
% 3. construct external potential
vext = getVharmonic(mol,1);
mol  = set(mol,'vext', vext);
% 4. set the number of electrons
mol = set(mol,'nel',4);
mol = set(mol,'spintype',2);
```

Fig. 11. Setting up a four-electron quantum dot

```
>> mol
Molecule: Quantum dot
   supercell:
      1.000e+01     0.000e+00     0.000e+00
      0.000e+00     1.000e+01     0.000e+00
      0.000e+00     0.000e+00     1.000e+01
   sampling size: n1 = 32, n2 = 32, n3 = 32
   atoms and coordinates: none
   number of electrons  : 4
   spin type            : 2
   kinetic energy cutoff: 2.500e+01
```

Fig. 12. Attributes of a four-electron quantum dot

For this problem, the SCF iteration appears to be more effective as we have already seen from Table VII.

```
Begin DCM calculation...
iterdcm = 1
resnrm =   5.439e-01
resnrm =   8.750e-01
resnrm =   1.075e+00
resnrm =   1.173e+00
   inner_scf = 1, Etot =  1.0467219866980e+01
   inner_scf = 2, Etot =  1.0466951470435e+01
   inner_scf = 3, Etot =  1.0466951470413e+01
------
iterdcm = 2
resnrm =   4.107e-01
resnrm =   6.688e-01
resnrm =   8.163e-01
resnrm =   9.329e-01
   inner_scf = 1, Etot =  1.0364915402274e+01
   inner_scf = 2, Etot =  1.0364915284577e+01
   inner_scf = 3, Etot =  1.0364915284577e+01
------
...
iterdcm = 49
resnrm =   1.752e-05
resnrm =   4.961e-05
resnrm =   5.023e-05
resnrm =   9.748e-05
   inner_scf = 1, Etot =  1.0133540758574e+01
   inner_scf = 2, Etot =  1.0133540758574e+01
   inner_scf = 3, Etot =  1.0133540758574e+01
------
iterdcm = 50
resnrm =   1.611e-05
resnrm =   3.769e-05
resnrm =   4.725e-05
resnrm =   8.170e-05
   inner_scf = 1, Etot =  1.0133540758308e+01
   inner_scf = 2, Etot =  1.0133540758308e+01
   inner_scf = 3, Etot =  1.0133540758308e+01
```

Fig. 13. Output from applying DCM to a four-electron quantum dot

7. CONCLUSIONS

We have described the design and implementation of KSSOLV, a MATLAB toolbox for solving the Kohn-Sham equations. Planewave discretization is used in KSSOLV because of its variational properties and simplicity. It is the natural choice for building a MATLAB Kohn-Sham equation solver because discrete Fourier transforms can be computed efficiently in MATLAB by its optimized FFT functions. The standard pseudopotential technique is utilized in KSSOLV to reduce the number of electron wavefunctions to be computed and the number of planewave basis functions required to represent each wavefunction. We should point out that planewave discretization does have some drawbacks [Kronik et al. 2006]. In particular, for molecules, the size of the fictitious supercell has to be large enough so that the computed wavefunctions do not overlap near the edge of the cell. However, these issues are not critical if one simply intends to study the existing algorithms or develop new algorithms for solving finite-dimensional Kohn-Sham equations.

One of the main features of the toolbox is the object-oriented design that allows users to easily set up a physical atomistic system. Physical attributes of the system are translated into numerical objects such as wavefunctions and Hamiltonians in a transparent fashion. These objects can be easily manipulated using overloaded algebraic operations. As a result, the coding effort required to investigate properties of the existing algorithms and to develop new algorithms for solving the Kohn-Sham equations is reduced significantly in KSSOLV. Furthermore, the visualization tools available in MATLAB enable users to quickly examine the computed results and compare the performance of different algorithms.

Some computational efficiency is sacrificed in KSSOLV to keep the object-oriented interface simple. To reduce the number of loops, which are slow in MATLAB, and replace them with vectorized operations (operations that can be applied simultaneously to all elements of an array), additional arrays are sometimes allocated to speed up the calculation. The presence of these arrays tends to increase the amount of memory usage. A number of improvements are being made to further reduce the memory and execution time required to construct and manipulate a Kohn-Sham Hamiltonian and electron wavefunctions. Alternative algorithms such as the Grassman manifold constrained total energy minimization scheme [Edelman et al. 1998] and the energy DIIS (EDIIS) algorithm [Kudin et al. 2006] will also be included in KSSOLV in future releases of the toolbox.

Because no parallelization has been implemented in the current version of KSSOLV, the size of the atomistic system one can study is rather limited. Nonetheless, many computational experiments can already be performed on the examples included in the package using the KSSOLV implementations of the SCF and DCM algorithms. We believe a great deal can be learned from running the existing codes on these examples.

Finally, because the atomic coordinates associated with each `Molecule` object can be easily modified in KSSOLV, the package can be easily extended in the future to facilitate structure relaxation (i.e. minimize the total energy with respect to both the atomic positions and single-particle wavefunctions) and *ab initio* molecular dynamics.

REFERENCES

Andreoni, W. and Curioni, A. 2000. New advances in chemistry and material science with CPMD and parallel computing. *Parallel Computing 26*, 819–842.

Arias, T. A., Payne, M. C., and Joannopoulos, J. D. 1992. Ab initio molecular dynamics: Analytically continued energ functionals and insights into iterative solutions. *Phys. Rev. Lett. 69*, 1077–1080.

Ashcroft, N. W. and Mermin, N. D. 1976. *Solid State Physics.* Brooks Cole, Pacific Grove, CA.

Baroni, S., Corso, A. D., de Gironcoli, S., Giannozzi, P., Cavazzoni, C., Ballabio, G., Scandolo, S., Chiarotti, G., Focher, P., Pasquarello, A., Laasonen, K., Trave, A., Car, R., Marzari, N., and Kokalj, A. 2006. PWscf. http://www.pwscf.org/.

Bekas, C., Kokiopoulou, E., and Saad, Y. 2005. Polynomial filtered lanczos iterations with applications in Density Functional Theory. *SIAM J. Matrix Anal. Appl. 30,* 1, 397–418.

Bloch, F. 1928. Über die Quantenmechanik der Elektronen in Kristallgittern. *Z. Physik 52*, 555–600.

Cancès, E. 2001. Self-consistent field algorithms for Kohn-Sham models with fractional occupation numbers. *J. Chem. Phys. 114*, 10616–10622.

Cancès, E. and Le Bris, C. 2000a. Can we outperform the DIIS approach for electronic structure calculations? *International Journal of Quantum Chemistry 79*, 82–90.

Cancès, E. and Le Bris, C. 2000b. On the convergence of SCF algorithm for the Hartree-Fock equations. *Math. Models. Numer. Anal. 34*, 749–7774.

Davis, P. J. 1979. *Circulant Matrices*. Wiley, New York.

Demmel, J. W. 1997. *Applied Numerical Linear Algebra*. SIAM, Philadelphia, PA.

Edelman, A., Arias, T. A., and Smith, S. T. 1998. The geometry of algorithms with orthogonality constraints. *SIAM J. Matrix Anal. Appl. 20,* 2, 303–353.

Ewald, P. P. 1921. Die Berchnung optischer und elektrostatischer Gitterpotentiale. *Ann. Phys. 64*, 253–287.

Gillan, M. J. 1989. Calculation of the vacancy formation in aluminum. *J. Phys. Condens. Matter 1*, 689–711.

Golub, G. H. and Van Loan, C. F. 1989. *Matrix Computations*, Third ed. Johns Hopkins, Philadelphia, PA.

Gonze, X., Beuken, J.-M., Caracas, R., Detraux, F., Fuchs, M., Rignanese, G.-M., Sindic, L., Verstraete, M., Zerah, G., Jollet, F., Torrent, M., Roy, A., Mikami, M., Ghosez, P., Raty, J.-Y., and Allan, D. 2002. First-principles computation of material properties : the ABINIT software project. *Computational Materials Science 25*, 478–492.

Hohenberg, P. and Kohn, W. 1964. Inhomogeneous electron gas. *Phys. Rev. B 136,* 3B, B864–B871.

Ihm, J., Zunger, A., and Cohen, M. L. 1979. Momentum-space formalism for the total energy of solids. *J. Phys. C: Solid State Physics 12*, 4409–4422.

Kerker, G. P. 1981. Efficient iteration scheme for self-consistent pseudopotential calculations. *Phys Rev. B 23*, 3082–3084.

Kleinman, L. and Bylander, D. M. 1982. Efficacious form for model pseudopotentials. *Phys. Rev. Lett. 48*, 1425.

Knyazev, A. 2001. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM J. Sci. Comput. 22,* 2, 517–541.

Kohn, W. and Sham, L. J. 1965. Self-consistent equations including exchange and correlation effects. *Phys. Rev. 140,* 4A, A1133–A11388.

Kresse, G. and Furthmüller, J. 1996. Efficiency of ab initio total energy calculations for metals and semiconductors using a plane-wave basis set. *Computational Materials Science 6*, 15–50.

Kresse, G. and Furthmüller, J. 1996. Efficient iterative schemes for *ab initio* total-energy calculations using a plane-wave basis set. *Phy. Rev. B 54*, 11169–11186.

Kronik, L., Makmal, A., Tiago, M. L., Alemany, M. M. G., Jain, M., Huang, X., Saad, Y., and Chelikowsky, J. R. 2006. PARSEC - the pseudopotential algorithm for real-space electronic structure calculations: Recent advances and novel applications to nano-structures. *Phys. Stat. Sol. 5*, 1063–1079.

Kudin, K. N., Scuseria, G. E., and Cances, E. 2006. A black-box self-consistent field convergence algorithm: One step closer. *J. Chem. Phys. 116,* 19, 8255–8261.

Le Bris, C. 2005. Computational chemistry from the perspective of numerical analysis. *Acta Numerica 14*, 363–444.

Mermin, N. D. 1965. Thermal properties of the inhomogeneous gas. *Phys. Rev. A 137*, 1441–1443.

Nogueira, F., Castro, A., and Marques, M. 2003. *A Primer in Density Functional Theory.* Springer, Berlin, Chapter A Tutorial on Density Functional Theory, 218–256.

NYQUIST, H. 1928. Certain topics in telegraph transmission theory. *Trans. AIEE 47*, 617–644.

PAYNE, M. C., TETER, M. P., ALLEN, D. C., ARIAS, T. A., AND JOANNOPOULOS, J. D. 1992. Iterative minimization techniques for *ab initio* total energy calculation: Molecular dynamics and conjugate gradients. *Reviews of Modern Physics 64,* 4, 1045–1097.

PERDEW, J. P. AND WANG, Y. 1992. Accurate and simple analytic representation of the electron-gas correlation energy. *Phys. Rev. B 45*, 13244–13249.

PERDEW, J. P. AND ZUNGER, A. 1981. Self-interaction correction to density-functional approximation for many-electron systems. *Phys. Rev. B 23*, 5048–5079.

PHILLIPS, J. C. 1958. Energy-band interpolation scheme based on a pseudopotential. *Phys. Rev. 112,* 3, 685–695.

PHILLIPS, J. C. AND KLEINMAN, L. 1958. New method for calculating wave functions in crystals and molecules. *Phys. Rev. 116,* 2, 287–294.

PICKETT, W. E. 1989. Pseudopotential methods in condensed matter applications. *Computer Physics Report 9*, 115–197.

PULAY, P. 1980. Convergence acceleration of iterative sequences: The case of SCF iteration. *Chemical Physics Letters 73,* 2, 393–398.

PULAY, P. 1982. Improved SCF convergence acceleration. *Journal of Computational Chemistry 3,* 4, 556–560.

RACZKOWSKI, D., CANNING, A., AND WANG, L. W. 2001. Thomas-Fermi charge mixing for obtaining self-consistency in density functional calculations. *Physical Review B 64*, 121101–1–4.

RITZ, W. 1908. Ueber eine neue methode zur lösung gewisser variationsproblem der mathematischen physik. *J. Reine Angew. Math 135*, 1–61.

SHAO, Y., MOLNAR, L. F., JUNG, Y., KUSSMANN, J., OCHSENFELD, C., BROWN, S. T., GILBERT, A. T., SLIPCHENKO, L. V., LEVCHENKO, S. V., ONEILL, D. P., JR, R. A. D., LOCHAN, R. C., WANG, T., BERAN, G. J., BESLEY, N. A., HERBERT, J. M., LIN, C. Y., VOORHIS, T. V., CHIEN, S. H., SODT, A., STEELE, R. P., RASSOLOV, V. A., MASLEN, P. E., KORAMBATH, P. P., ADAMSON, R. D., AUSTIN, B., BAKER, J., BYRD, E. F. C., DACHSEL, H., DOERKSEN, R. J., DREUW, A., DUNIETZ, B. D., DUTOI, A. D., FURLANI, T. R., GWALTNEY, S. R., HEYDEN, A., HIRATA, S., HSU, C.-P., KEDZIORA, G., KHALLIULIN, R. Z., KLUNZINGER, P., LEE, A. M., LEE, M. S., LIANG, W. Z., LOTAN, T., NAIR, N., PETERS, B., PROYNOV, E. I., PIENIAZEK, P. A., RHEE, Y. M., RITCHIE, J., ROSTA, E., SHERRILL, C. D., SIMMONETT, A. C., SUBOTNIK, J. E., III, H. L. W., ZHANG, W., BELL, A. T., CHAKRABORTY, A. K., CHIPMAN, D. M., KEIL, F. J., WARSHEL, A., HEHRE, W. J., III, H. F. S., KONG, J., KRYLOV, A. I., GILL, P. M. W., AND HEAD-GORDON, M. 2006. Advances in methods and algorithms in a modern quantum chemistry program package. *Phys. Chem. Chem. Phys. 8*, 3172–3191.

TETER, M. P., PAYNE, M. C., AND ALLAN, D. C. 1989. Solution of Schrödinger's equation for large systems. *Physical Review B 40,* 18, 12255–12263.

TREFETHEN, L. N. AND BAU III, D. 1997. *Numerical Linear Algebra*. Siam, Philadelphia, PA.

TROULLIER, N. AND MARTINS, J. L. 1991. Efficient pseudopotentials for plane-wave calculations. *Phys. Rev. B 43*, 1993–2005.

VAN LOAN, C. 1987. *Computational Frameworks for the Fast Fourier Transform*. SIAM, Philadelphia, PA.

VANDEVONDELE, J. AND HUTTER, J. 2003. An efficient orbital transformation method for electronic structure calculations. *Journal of Chem. Phys. 118*, 4365–4369.

VOORHIS, T. V. AND HEAD-GORDON, M. 2002. A geometric approach to direct minimization. *Molecular Physics 100,* 11, 1713–1721.

WANG, L. 2008. PETOT. http://hpcrd.lbl.gov/ linwang/PEtot/PEtot.htm.

WEINERT, M. AND DAVENPORT, J. W. 1992. Fractional occupations and density-functional energies and forces. *Phys. Rev. B 45*, 13709–13712.

WNTZCOVITCH, R. M., MARTINS, J. L., AND ALLEN, P. B. 1992. Energy versus free-energy in first-principles molecular dynamics. *Phys. Rev. B 45*, 11372–11374.

YANG, C. 2007. KSSOLV User's Guide. Tech. Rep. LBNL-63661, Lawrence Berkeley National Laboratory.

Yang, C., Meza, J. C., and Wang, L. W. 2005. A constrained optimization algorithm for total energy minimization in electronic structure calculation. *Journal of Computational Physics 217*, 709–721.

Yang, C., Meza, J. C., and Wang, L. W. 2007. A trust region direct constrained minimization algorithm for the Kohn-Sham equation. *SIAM J. Sci. Comp. 29,* 5, 1854–1875.

Yin, M. T. and Cohen, M. L. 1982. Theory of *ab initio* pseudopotential calculations. *Phys. Rev. B 25,* 12, 7403–7412.

Zhou, Y., Saad, Y., Tiago, M. L., and Chelikowsky, J. R. 2006. Self-consistent field calculations using Chebyshev-filtered subspace iteration. *J. of Comp. Phys. 219*, 172–184.

...